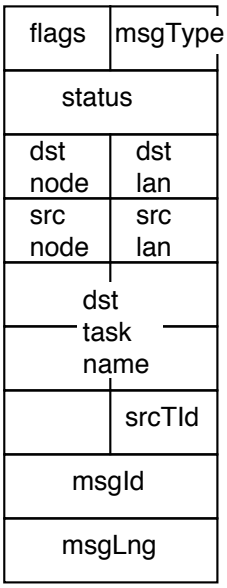
Simple Protocol for SRMs

Protocol #4

Sep 26, 1990

For VME Local Station communications with the SRM arcnet nodes, a choice of protocol must be made. One can use an existing protocol already known to the Local Station, or one can invent a new one designed for the purpose. This option—called "#4" in our informal discussions due to the existence of support already for the Classic, D0 and Accelerator protocols—should be simple, or it would not be worth the effort. An idea for a suitable protocol is explored in this note.

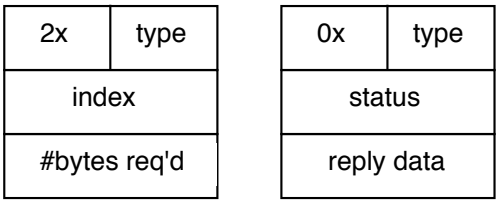
As an aid to get started, assume we use the Acnet header as a basis for a simple protocol design. It is well-known around the accelerator division and provides for expandable and generic task-to-task communications. It allows both one-shot and repetitive replies to generic requests. For reference, its layout is repeated here:



The `msgType` can be a Request, a Reply, or a USM (unsolicited message). The Request demands a reply. The USM demands no reply. The destination task name for a request or USM allows designing a large number of non-interfering protocols, since only the tasks involved in the communication must understand the protocol used in the rest of the message beyond the header. The source task id provides for routing the reply back to the requester. Multiple requests between tasks are distinguished by the `msgId`. The `msgLng` gives the entire message length including the Acnet header (18 bytes) itself. A flag bit in the `msgType` byte indicates whether a request expects a single reply or multiple replies. Network Layer software supports the use of the Acnet header to provide the task-to-task communications. A task connects to the network to announce its support for handling requests destined for a given destination task name and provides a message queue that enables it to receive such requests and any replies to its own requests. (Note that a task name is here not the same as a task name known to the operating system kernel.)

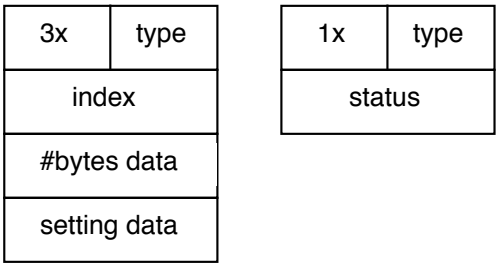
Additional items needed in a simple message protocol for data requests and settings are a message type (beyond the generic `msgType` mentioned above), a device index and either the #bytes of data requested or the setting data.

Consider the following layout for a data request and reply:



The value of "`x`" is the length of the index value. This would be 2 for channel or bit numbers and 4 for memory addresses. The `type` byte can denote analog data, binary status or memory data. The reply can include the same value used in the request.

Consider the following formats for a setting and its acknowledgment:



Again the "`x`" nibble gives the size of the index value. The #bytes of setting data is included in order to allow grouped settings. Without this consideration, it can be inferred from the `msgLng` word in the header.

Whether support for this simple protocol is worth the effort is yet to be decided.